PTT-2025-031 – Sensitive File Read via Path Traversal

CVE-ID: CVE-2026-30462

Credits:

Matei “Mal” Bădănoiu and Raul Bledea of Pentest-Tools.com

Environment:

- FuelCMS v1.5.2

CVSSv3: 4.3 Medium - AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

Note: In case the attacker recovers valid DB credentials and can connect directly to the DB the following changes occur:

- **Confidentiality** increases to: **high**. The attacker can extract all information contained by the website via direct DB queries.
- **Integrity** increases to: **high**. The attacker can forcefully reset an admin’s password or make another user an admin and modify or delete any element within the website.
- **Availability** increases to: **high**. The attacker can modify or delete critical parts of the DB resulting in the unavailability of the FuelCMS application.

8.8 High - AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Description:

An authenticated user can read specific files (need to have the extensions “.php”) on the target system (as the “www-data” user) by exploiting a path traversal vulnerability in the "Blocks" module. This is possible because FuelCMS doesn't properly sanitize the “Block View File” names allowing path traversal.

Requirements:

In order to perform this exploit a user would require to be associated with a role that allows “POST” requests on the “/fuel/blocks/edit/” endpoint.

No Fix:

The FuelCMS software master branch has seen *no updates* in ~4 years. Although we emailed the vendor, we do not expect this vulnerability to ever be fixed.

Proof of Concept:

The following HTTP “POST” request can be used to leverage path traversal to force the “Blocks” module to load arbitrary PHP files present on the system.

Note: The “Blocks” component is limited to only exfiltrating PHP files as the “.php” extension is added automatically.

This behaviour can be used to exfiltrate sensitive files such as:

- “database.php” which contains DB connection information such as the host, user and password
- “config.php” that contains FuelCMS configuration settings and may contain other sensitive information

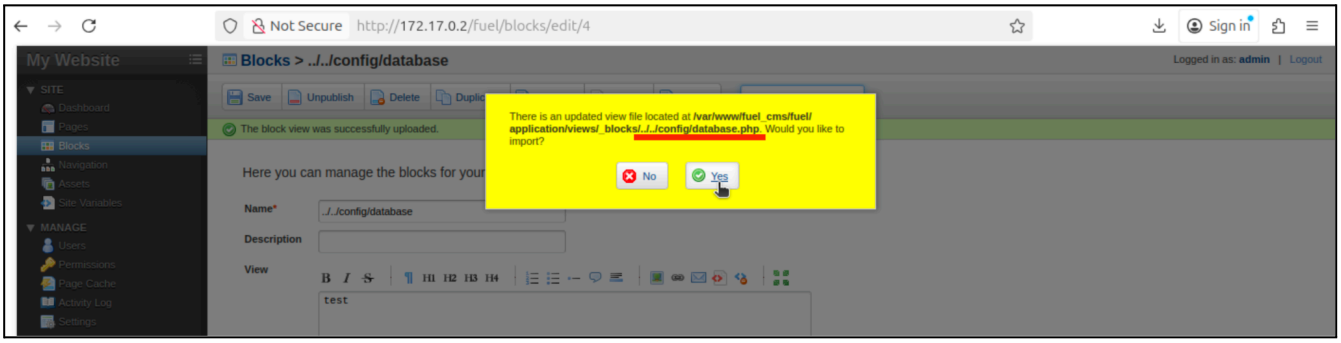
To perform the exploit the following steps are required:

1. Open the “Blocks” feature and select to “Upload” a new block:

Note: As in this case we want to exfiltrate the “database.php” file we will insert into the “Name” field the value “../../config/database”.

Note 2: The uploaded “test.txt” does not contain any relevant information necessary for the exploit, but is required by the “Upload” function.

2. After clicking the “Yes, upload it” button we will be redirected and, if the PHP file exists on the system, we will be prompted if we would like to “update”/“import” the file’s content:



Note: We can also observe the full path where the application is installed on the server in the server’s response.

3. After clicking “Yes” the content of the “database.php” file should be returned in the “View” field: